Operating System and Design (19CS2106A)
Advanced Lab- 2

XV6 design, implementation, and customization.

1.TOUCH

STEP1: Open Vi Editor
Syntax : vi touchex.c

STEP2: Type the below code(Press 'i' to enter into insert mode

```
#include "types.h"
#include "stat.h"
#include "user.h"
#include "fcntl.h"
#include "fs.h"
int main(int argc,char *argv[])
{
   if(argc<2)
   {
      printf(1,"Usage: touch [files]...\n");
      exit();
   }
   int i,err;
   for(i=1;i<argc;i++)
   {
      if((err=open(argv[i],O_CREATE|O_RDWR)) < 0)
      {
         printf(1,"touch: error where creating %s\n",argv[i]);
         exit();
      }
      close(err);
   }
   exit();
}
```

STEP 3: Press Esc : wq to save and quit from the editor after typing the program.

STEP 4: Open Makefile
Syntax: vi Makefile

STEP 4: IN Makefile program do the following changes in two sections:

In the Makefile, there are two places in which we need to put entries.
Find the place with some lines like the following.
We have to add a line as shown below to notify about our new program.

```
UPROGS= \
                    _cat\
                    _echo\
                    _forktest\
                    _grep\
                    _init\
                    _kill\
                    _ln\
                    _ls\
                    _mkdir\
                    _rm\
                    _sh\
                    _stressfs\
                    _usertests\
                    _wc\
                    _zombie\
                    _touchex\
```

Similarly, find the place with the lines like below.
Add an entry as shown to indicate that we have a program called my.c there.

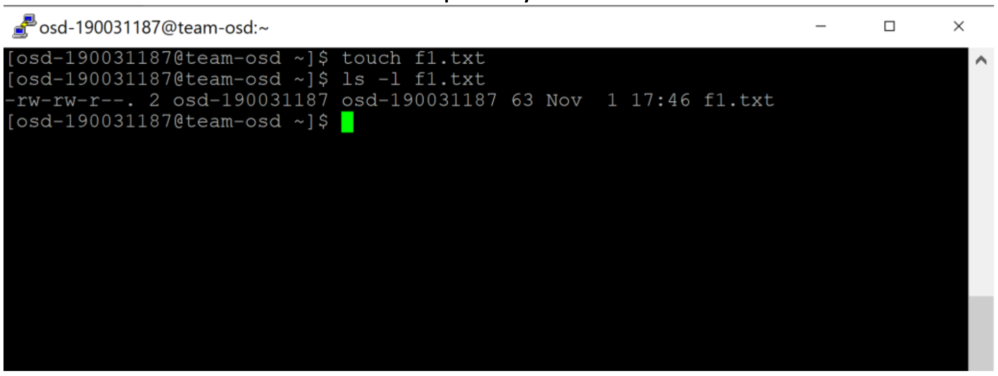
```
EXTRA=\ mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.c kill.c\
ln.c ls.c mkdir.c rm.c stressfs.c usertests.c wc.c zombie.c\ touchex.c\
printf.c umalloc.c\ README dot-bochsrc *.pl toc.* runoff runoff1 runoff.list\ .gdbinit.tmpl
gdbutil\
```

Now, our Makefile and our user program is ready to be tested.
Enter the following commands to compile the whole system.

Syntax:
```
make clean
make
```
Now, start xv6 system on QEMU and when it booted up, run ls command
to check whether our program is available for the user.

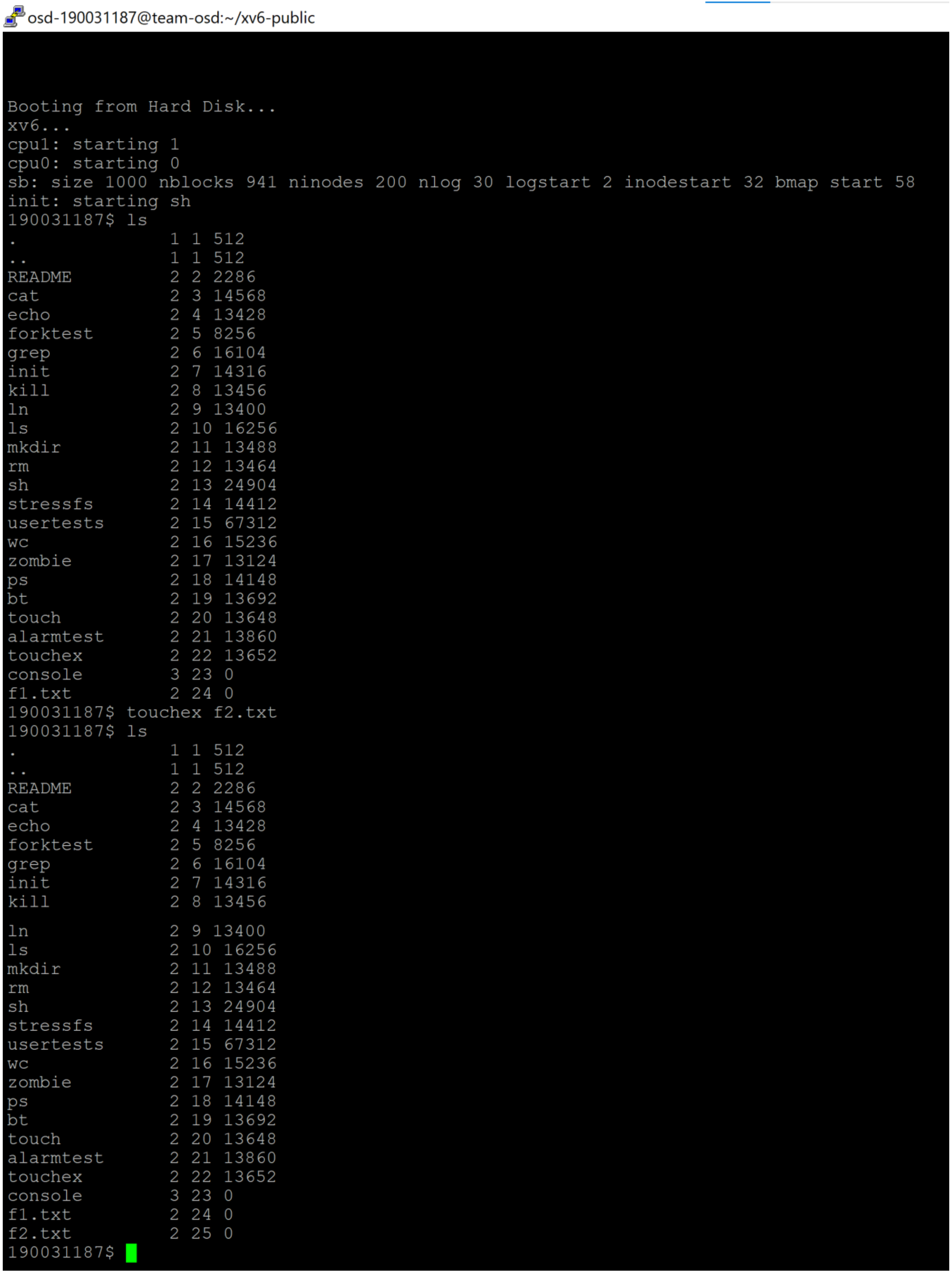
Syntax:
```
make qemu-nox
```

```
$ls
```
Check whether touchex is listed in the output.If yes then use that as a command.



```
[osd-190031187@team-osd ~]$ touch f1.txt
[osd-190031187@team-osd ~]$ ls -l f1.txt
-rw-rw-r--. 2 osd-190031187 osd-190031187 63 Nov  1 17:46 f1.txt
[osd-190031187@team-osd ~]$
```

OUTPUT



osd-190031187@team-osd:~/xv6-public

```
Booting from Hard Disk...
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
190031187$ ls
.              1 1 512
..             1 1 512
README         2 2 2286
cat            2 3 14568
echo           2 4 13428
forktest       2 5 8256
grep           2 6 16104
init           2 7 14316
kill           2 8 13456
ln             2 9 13400
ls             2 10 16256
mkdir          2 11 13488
rm             2 12 13464
sh             2 13 24904
stressfs       2 14 14412
usertests      2 15 67312
wc             2 16 15236
zombie         2 17 13124
ps             2 18 14148
bt             2 19 13692
touch          2 20 13648
alarmtest      2 21 13860
touchex        2 22 13652
console        3 23 0
f1.txt         2 24 0
190031187$ touchex f2.txt
190031187$ ls
.              1 1 512
..             1 1 512
README         2 2 2286
cat            2 3 14568
echo           2 4 13428
forktest       2 5 8256
grep           2 6 16104
init           2 7 14316
kill           2 8 13456

ln             2 9 13400
ls             2 10 16256
mkdir          2 11 13488
rm             2 12 13464
sh             2 13 24904
stressfs       2 14 14412
usertests      2 15 67312
wc             2 16 15236
zombie         2 17 13124
ps             2 18 14148
bt             2 19 13692
touch          2 20 13648
alarmtest      2 21 13860
touchex        2 22 13652
console        3 23 0
f1.txt         2 24 0
f2.txt         2 25 0
190031187$ 
```

2) TAIL COMMAND IN XV6:

STEP1: Open Vi Editor
Syntax : vi tailex.c

STEP2: Type the below code(Press ‘i’ to enter into insert mode)

```
#include "types.h"
#include "stat.h"
#include "user.h"

char buf[1024]={'\\'};//Initialise buffer1
char buf2[1024]={'\\'};//Initialise buffer2

void tail(int fd,char *name, int x)
{
        int i,n,m,l;
        int tot_lines;
        tot_lines=0;
        int start;

        while((n=read(fd,buf,sizeof(buf)))>0)
        {
                for(i=0;i<=n ;i++)
                {
                        if(buf[i]=='\n')
                        {
                          tot_lines++;  // Loop for total number of lines in the file
                          if(strcmp(name,"")==0){
                                printf(1,"\n");}
                        }
                        else
                        {
                        if(strcmp(name,"")==0){
                                if(buf[i] =='\0')  // Check end of file
                                {
                                        exit();
                                }
                                if(buf[i]!='\n')
                                {
                                 printf(1,"%c",buf[i]);
                                }
                                else
                                {
                                 printf(1,"\n");}
                                }
```

```
                    }
                }

        }
        close(fd);
        start = tot_lines-x;
        l=0;
        int fd2 = open(name,0); //Creating file descriptor 2
        while((m=read(fd2,buf2,sizeof(buf2)))>0)
        {
                for(i=0;i<=m;i++)
                {
                        if(buf2[i] == '\n')
                        l++;
                        if(l>=start)
                        {
                                if(buf2[i] !='\n' && l>=start)
                                {
                                        printf(1,"%c",buf2[i]);
                                }
                                else
                                {
                                        printf(1,"\n");
                                        l++;
                                }
                        }
                }
        }
        close(fd2);
        if(n<0)
        {
                printf(1,"tail: error while reading \n");
                exit();
        }
}

int
main(int argc,char *argv[])
{
        int fd,i;

        if(argc<=1)
        {
                tail(0,"",10);
                exit();
        }
```

```
	else if(argc==2)
	{
		for(i=1;i<argc;i++)
		{
			if((fd=open(argv[i],0))<0 )
			{
				printf(1,"Error in File Reading");
				exit();

			}
			tail(fd,argv[i],10);

			close(fd);
		}

	}
	else if(argc==3)
	{
		char c[1024];
		strcpy(c,argv[1]);
		char *num_str=c;
		num_str=num_str+1;
		int x= atoi(num_str);

		for(i=2;i<argc;i++)
		{
			if((fd=open(argv[i],0))<0 )
			{
				printf(1,"tail:error opening the %s\n",argv[i]);
				exit();
			}
			tail(fd,argv[i],x);
			close(fd);
		}
	}

	else
	{
		printf(1,"tail: syntax error");
	}
	exit();
}
```

STEP 3: Press Esc : wq to save and quit from the editor after typing the program.

STEP 4: Open Makefile
Syntax: vi Makefile

STEP 4: IN Makefile program do the following changes in two sections:

In the **Makefile**, there are two places in which we need to put entries. Find the place with some lines like the following. We have to add a line as shown below to notify about our new program.

```
UPROGS= \
                                   _cat\
                                  _echo\
                                  _forktest\
                                  _grep\
                                  _init\
                                  _kill\
                                  _ln\
                                  _ls\
                                  _mkdir\
                                  _rm\
                                 _sh\
                                  _stressfs\
                                  _usertests\
                                  _wc\
                                  _zombie\
                                  _tailex\
```

Similarly, find the place with the lines like below. Add an entry as shown to indicate that we have a program called **my.c** there.

EXTRA=\ mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.c kill.c\ ln.c ls.c mkdir.c rm.c stressfs.c usertests.c wc.c zombie.c\ tailex.c

printf.c umalloc.c\ README dot-bochsrc *.pl toc.* runoff runoff1 runoff.list\ .gdbinit.tmpl gdbutil

Now, our Makefile and our user program is ready to be tested. Enter the following commands to compile the whole system.

Syntax:

```
make clean
make
```

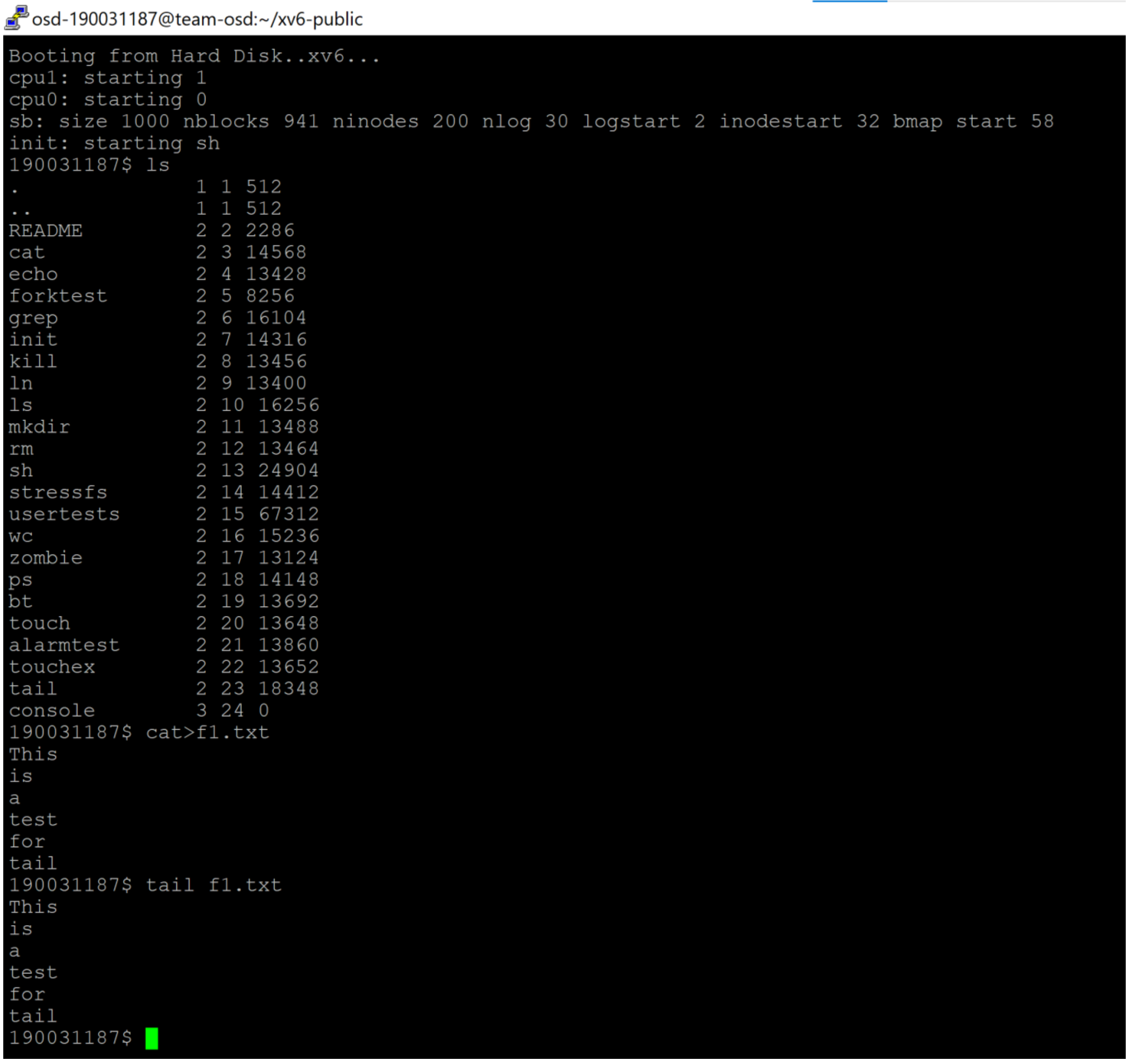
Now, start xv6 system on QEMU and when it booted up, run ls command to check whether our program is available for the user.

Syntax:

```
make qemu-nox
$ls
```

Check whether touchex is listed in the output.If yes then use that as a command.
$tailex f1.txt
Output:
Last 10 lines of the file f1.txt will be displayed by default

OUTPUT


osd-190031187@team-osd:~/xv6-public

```
Booting from Hard Disk..xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
190031187$ ls
.              1 1 512
..             1 1 512
README         2 2 2286
cat            2 3 14568
echo           2 4 13428
forktest       2 5 8256
grep           2 6 16104
init           2 7 14316
kill           2 8 13456
ln             2 9 13400
ls             2 10 16256
mkdir          2 11 13488
rm             2 12 13464
sh             2 13 24904
stressfs       2 14 14412
usertests      2 15 67312
wc             2 16 15236
zombie         2 17 13124
ps             2 18 14148
bt             2 19 13692
touch          2 20 13648
alarmtest      2 21 13860
touchex        2 22 13652
tail           2 23 18348
console        3 24 0
190031187$ cat>f1.txt
This
is
a
test
for
tail
190031187$ tail f1.txt
This
is
a
test
for
tail
190031187$ 
```

UNIX system programming

1.lseek: Positioning the Offset

```
#include “param.h”
#include "types.h"
#include "stat.h"
#include "user.h"
#include "fs.h"
#include "fcntl.h"
#include "syscall.h"
#include "traps.h"
#include "memlayout.h"
int
main(int argc, char *argv[]) {
int fp;
char *buffer = 0;
uint len;
if (argc != 5) {
printf(1, "usage: ./lseek1 <filename> <offset>\<len> <string>\n");
exit();
}
if ((fp = open(argv[1], O_RDONLY)) < 0) {
 printf(1, "unable to open file %s\n", argv[1]);
 exit();
}
len = atoi(argv[3]);
if ((buffer = (char *)malloc(len + 1)) < 0) { printf(1, "unable to allocate buffer\n");
exit();
}
int offset = atoi(argv[2]);
 int ret;
ret = lseek(fp, SEEK_SET, offset);
if (ret < 0) {
printf(1, "unable to lseek\n");
exit();
}
read(fp, buffer, len); buffer[len] = 0;
printf(1, "(%s:%s)\n", argv[4], buffer);
if (strcmp(buffer, argv[4])) { printf(1, "strings do not match\n");
exit();
}
printf(1, "strings match\n");
exit();
}
```

Step 1: nano fcntl.h,
add the following

```
#define SEEK_SET 0x001
#define SEEK_CUR 0x002
#define SEEK_END 0x003
```

Step 3: open syscall.c and add

```
extern int sys_lseek(void)
[SYS_lseek] sys_lseek,
```

Step 4: open syscall.h and add

```
#define SYS_lseek 22
```

Step 5: open sysfile.c and add the following code

```
uint sys_lseek(void)
{
struct file *f;
int offset;
uint whence;
if (argfd(0, 0, &f) < 0 || argint(2, &offset) < 0 || argint(1, (int *)&whence) < 0)
return -1;
if (f->type != FD_INODE)
return 0;
uint offset_temp; uint filesize;
ilock(f->ip);
filesize = f->ip->size;
iunlock(f->ip);
switch(whence) {
case SEEK_SET: offset_temp = 0;
break;
case SEEK_CUR: offset_temp = f->off;
break;
case SEEK_END: offset_temp = filesize;
break;
default: return -1; break; } // xv6 read and write do not account for 'holes'
// so better not allow exceeding the bounds
if (((offset_temp + offset) < 0 ) || ((offset_temp + offset) >=filesize))
return -1;
f->off = offset_temp + offset;
return f->off;
}
```

Step 6: open user.h and add

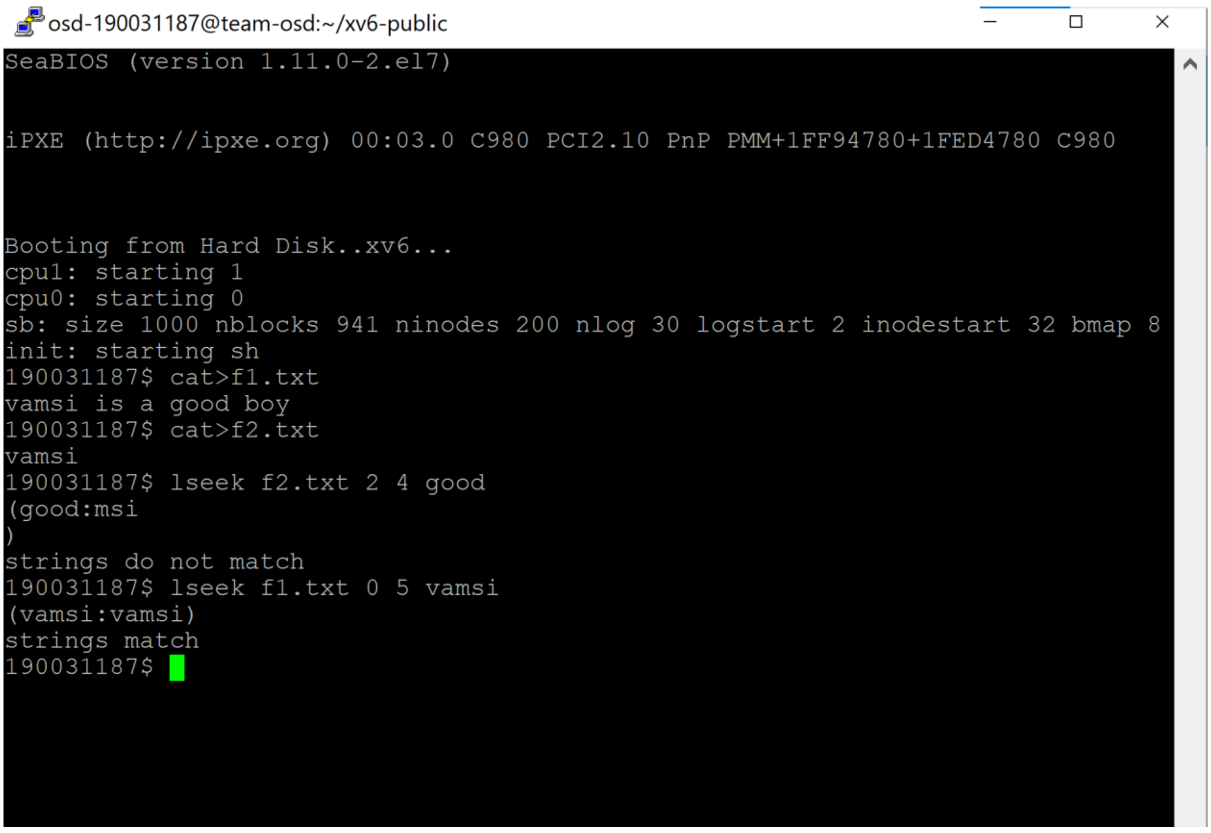
```
int lseek(int fd, int offset, int whence);
```

Step 7: open usys.S and add
SYSCALL(lseek)

Step 8: add _lseek in Uprogs and lseek.c in Extras in Makefile

Step 9: make qemu-nox

OUTPUT



```
SeaBIOS (version 1.11.0-2.el7)


iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+1FF94780+1FED4780 C980



Booting from Hard Disk..xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap 8
init: starting sh
190031187$ cat>f1.txt
vamsi is a good boy
190031187$ cat>f2.txt
vamsi
190031187$ lseek f2.txt 2 4 good
(good:msi
)
strings do not match
190031187$ lseek f1.txt 0 5 vamsi
(vamsi:vamsi)
strings match
190031187$ 
```

2. Pointerreverse_read.c: Reading a File in Reverse

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, const char * argv[]) {

    FILE *file;

    file = fopen("./oz.txt", "r+");
    if (file == NULL)
    {
        printf ("Error - Couldn't open file\n");
    }

    fseek(file, 0, SEEK_END); // move file pointer to end of file
    long size = ftell(file); // file pointer position == character count in file
    fseek(file, 0, SEEK_SET);  // move back to beginning of file

    char* buffer = (char*) malloc(size * sizeof(char));

    fread(buffer, sizeof(char), size, file); // read file contents to buffer

    for(long i = 0; i < size/2; ++i)
    {
        buffer[i] = buffer[size-i-1];
    }

    fseek(file, 0, SEEK_SET);

    fwrite(buffer, sizeof(char), size, file); // Write reverted content


    free(buffer);
    fclose(file);

    return 0;
}
```